14-05-2023

Introduction to Passkeys Usage and Implementation

GN5-1 Trust and Identity Incubator

Table of Contents

1 Overview

Passkeys are a more secure and convenient alternative to passwords, aimed at becoming the primary factor for account authentication and provide a frictionless login experience while preventing the theft or guessing of secrets. They combine secure authentication standards developed by the FIDO Alliance and the W3C Web Authentication Working Group, are supported by major vendors such as Google, Apple and Microsoft and are already available on many websites.

Passkeys validate the user against locally and securely stored credentials on smartphones and laptops. The user must provide a valid PIN, password, or biometric feature such as face or fingerprint and may sign in to one device with another.

They can be used in step-up authentication or transaction confirmation to perform critical operations or access sensitive data. They can also be used in enhanced security scenarios where one or more passkeys or additional factors verify the user's presence and identity. Using passkeys reduces the need for other secondary authentication methods such as SMS or app-based one-time codes.

Although passkeys are already supported by major platforms, a full transition from passwords to passkeys requires all service providers to invest time, effort, and money to change their services and websites. Therefore, solutions and guidelines supporting the implementation of passkeys could significantly speed up this transition.

As a part of step-up authentication or transaction confirmation, passkeys can be used after the user has already signed in and wants to perform a critical operation or access particularly sensitive data. The user is requested to complete an additional step to more strongly confirm their identity before accessing a certain service or completing a transaction.

Passkeys can be also used in enhanced security scenarios, where one or more passkeys or additional factors can additionally ensure that the user has been verified and their presence tested.

According to the common understanding adopted in this document, passkeys are FIDO/WebAuthn public key credentials that can be discovered on devices during sign-in, that is, selected by the user among those applying for the Relying Party. They also require user verification and can usually be replicated to other user devices and backed up. These features enable passkeys to be integrated into credentials management platforms and browsers and support common user flows. Additionally, single-device passkeys are available only from one device and cannot be copied.

2 How do passkeys work?

Passkeys work seamlessly across operating systems and browsers and can be used on websites as well as in native applications. They are suitable for signing in in common service scenarios and for more sensitive accounts, such as those with banks and other financial services. Passkeys offer several mechanisms that greatly enhance and integrate the user experience. This is critical for their wider adoption as a means of passwordless authentication.

Passkeys are a more secure authentication method than passwords which are prone to human error. They are not generated or used in clear text, their secret private key information is not visible to the user, and are not directly managed by users. Thus, they cannot inadequately create, use with several or wrong Relying Parties, or share them. Passkeys are not transmitted during sign-in and cannot be stolen in a breach or tracked between websites. They provide robust protection against phishing attacks and do not expose the user's biometric data. Passkeys require close proximity to the

authentication device, unlike passwords which can be used or provided from arbitrary locations or devices.

2.1 Architecture

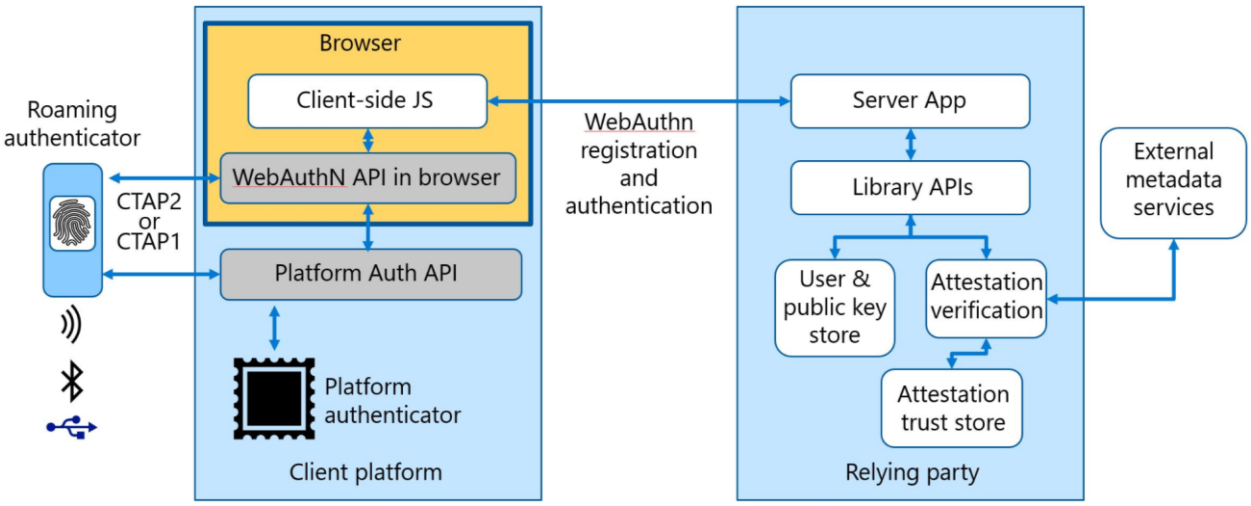

Figure 1. Passkeys architecture: FIDO2 (CTAP) and WebAuthn (source: https://twitter.com/john_craddock/status/1224620080924971008)

Several components work together to provide a secure and user-friendly authentication mechanism with passkeys for online services and applications:

- The **roaming authenticator** (or external authenticator) is a FIDO2 authenticator that is not physically attached to the user's device. It is a standalone device that can be connected to the user's device via Bluetooth, USB or other method. With passkeys, a smartphone can also serve as a roaming authenticator.
- The **client platform** is the user's device, such as a smartphone or laptop, on which the authentication process is initiated and the client side of the application is executed within the browser or native application.
- The **browser** allows the user to access websites and web applications and provides the WebAuthn API for web applications to communicate with authenticators.
- **Client-side JS** is JavaScript code that runs within the browser and interacts with the WebAuthn API, authenticators and the Relying Party to implement registration and sign-in passkey flows. A native application can be written in any supported language.
- The **WebAuthn API** is a browser-based software interface used to generate and use passkeys during registration and sign-in. If a native application is used, a platform-specific API such as Apple WebKit or Google Credential Manager can be used instead of WebAuthn.
- The **platform authentication API** enables the use of built-in authentication methods, such as fingerprint or facial recognition, as well as biometric sensors and secure elements available on user devices.
- The **platform authenticator** is a FIDO2 authenticator built into the user's device's operating system or firmware. It typically includes a Trusted Platform Module (TPM) and biometric sensors.
- The **Relying Party** is the online service that requires authentication to access its resources. It interacts with the browser using HTTPS.
- The **server application** runs on the Relying Party's server and receives authentication requests from the client platform.
- **Library APIs** are software interfaces that allow developers to integrate authentication into their applications.

- The **user and public key store** is a secure storage location for the user's private keys and the public keys associated with their passkeys.
- **Attestation verification** is the process of verifying the authenticity of a FIDO2 authenticator. It involves checking the attestation statement provided by the authenticator to ensure it was signed by a trusted party.
- The **attestation trust store** is a database of trusted attestation roots and intermediate certificates used to verify the authenticity of FIDO2 authenticators.
- **External metadata services** are servers that provide additional information about FIDO2 authenticators, such as their capabilities and supported algorithms, to Relying Parties during the authentication process.

2.2 WebAuthn and FIDO2 standards

To implement passkeys, websites and remote applications use WebAuthn, while client operating systems and passkey-sharing platforms use the FIDO2 CTAP2 API to access locally stored credentials.

W3C Web Authentication (WebAuthn) is a web standard published by the World Wide Web Consortium (W3C) [https://www.w3.org/TR/webauthn-3/, https://w3c.github.io/webauthn/]. **WebAuthn Level 1** is a W3C Recommendation published in 2019, and it is supported by Chrome, Firefox, Edge, Safari and Opera. The **WebAuthn Level 2** standard was published as a W3C Recommendation in April 2021. It provides fixes and improvements to the original specification that extend functionality for specific use cases.

The FIDO Alliance and the W3C published the **FIDO2 Web Authentication** standard specification in 2018. FIDO2 [https://fidoalliance.org/fido2/] enables users to use common devices to easily authenticate their identity for online services in both mobile and desktop environments. By adding the FIDO Client to Authenticator Protocol 2 (CTAP2) to WebAuthn, CTAP2 enables a FIDO2-compliant cryptographic authenticator to interoperate with a WebAuthn client.

3 Usage

A passkey is used to identify a user account with a particular service. It can be kept on a smartphone, tablet, PC, or a password manager. The passkey may be shared among a set of devices or can be bound to a specific device. A passkey ensures a strong and private relationship between a person and a website or application. Typically, users have different passkeys for different services, unless these services are provided by the same Relying Party. However, it is possible to have multiple passkeys for the same account, when the browser will display them all so that the user could choose which one to use.

The passkey hides the mechanism used to verify the user, which is typically tied to biometrics or a PIN and a device or physical key. The mechanism on the device includes private user verification data, with the private key usually stored on a secured module of the user's device or a roaming hardware authenticator. In addition, there is software tied to the operating system and ecosystem supporting passkeys, which provide usage and synchronisation across devices. This greatly increases convenience and ease of use and provides a higher level of reliability than using passwords, which can also be handled by the platform. However, recovering a passkey if the user's last device is lost requires the passkey's private key to be stored in a cloud of the operating system or platform vendor, which, along with sharing among several devices, introduces security risks and may affect the Level of Assurance (LoA) of the passkey.

Even browsers that do not support passkeys by default can get an extension that comes with its ecosystem. For example, Dashlane can integrate passkeys across Safari, Firefox, Chrome, Edge and (soon) mobile devices. OS-integrated or standalone password managers (that is, credential management applications) are rapidly evolving into solutions that manage passkeys in a user-friendly way, linking web browsers, devices, user identities and services with passkeys.

3.1 Registration

When creating an account for a website or other resource provider that supports passkeys, the user undergoes a registration procedure and is offered the option to create a passkey or sign in with a security device. A passkey is generated on the user's device for that particular website. The public key is sent to the website's server for storage, while the private key remains securely stored in the device's authenticator.

The private cryptographic key of a passkey persists only on the user's devices, such as laptops or mobile phones, or in a credential store of a detachable secure key. It is safeguarded by biometrics and hardware-based security, if present and used. The associated public key is stored by the accessed service on the application server or by the application, while the private key is never sent to the service.

3.2 Authentication

The authentication process involves the user visiting a service, receiving a randomly generated challenge from the server, selecting and using a passkey with the help of their browser, operating system or password manager, and confirming their identity by unlocking their device using a biometric sensor, PIN or pattern. The user's device then uses the private key kept in the device's authenticator to sign the challenge, and the service verifies the authenticity of the signature using the user's public key. If the signature is valid, the user is granted access to the service.

3.3 Use across devices

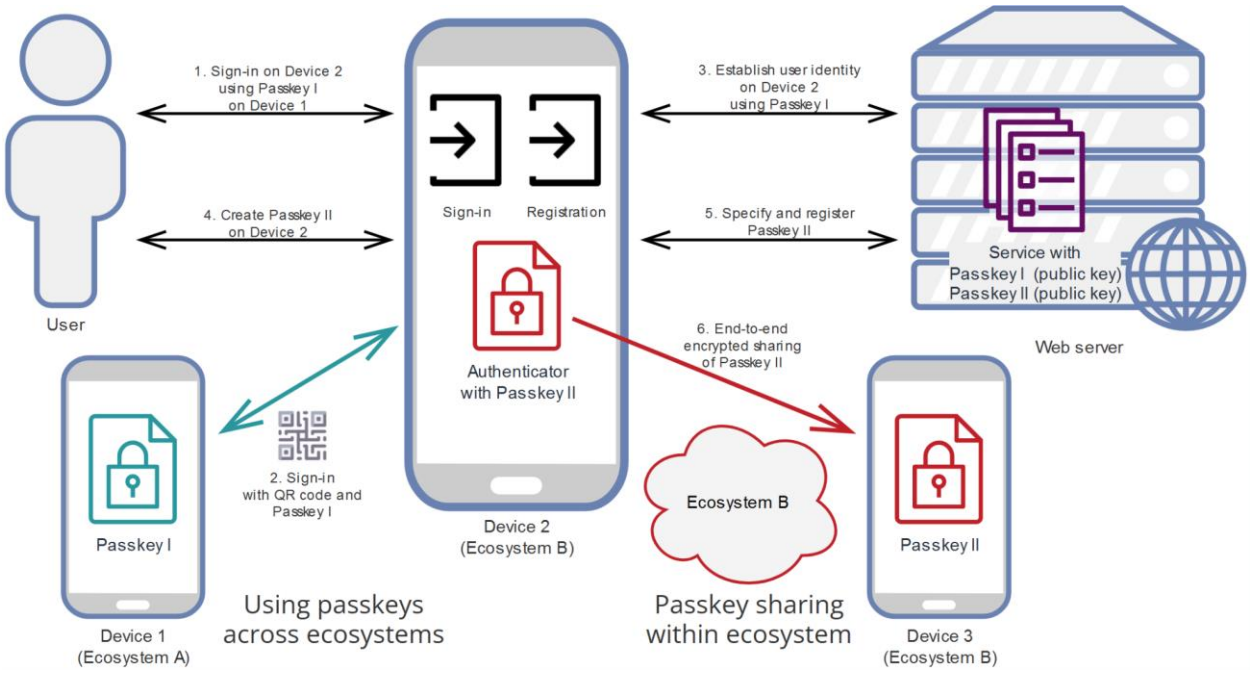

Figure 3: Using an existing passkey to sign in at another device and create a passkey for its ecosystem

Users can sign on to their computer by using a passkey created on and stored on their mobile device. This cross-device and cross-platform authentication is standardised by FIDO, involves scanning a QR code with the smartphone and requires Bluetooth to ensure physical proximity between devices to prevent phishing. After successful authentication, a new passkey may be created for the device accessing the service, and this passkey will be shared with all other devices in the device's ecosystem. The passkey is not transmitted to the computer during the authentication process.

Since the registration, sign in, passkey sharing and backup are mediated by the browser, the operating system and sharing platform, service and application developers do not need to know anything about QR codes, Bluetooth or CTAP. The described interactions are demonstrated and described in [https://www.youtube.com/watch?v=SWocv4BhCNg].

For more detailed information about support on different platforms and with different browsers, go to [https://passkeys.dev/device-support/].

4 Implementation

There are two ways to implement passkey sign-in. The first way suitable for applications that only have passkey users, is by using the "Sign in with a passkey" button. This allows users to sign in to a website or application without filling out forms. This is a "modal UI" where a selector displays the applicable passkeys for the user to choose from. However, for applications that already have password-based or multifactor authentication in place, the website must still offer a sign-in form with username and password fields. The second way, passkey autofill or "conditional UI", is suitable for transitioning from password-based to passkey-based authentication. This automatically shows suggestions for suitable passwords and passkeys through autofill. If the account is passkey-based, the user is immediately prompted to unlock the device to sign in. In both cases, the user can also choose to use a passkey on another device by scanning a QR code or using a security key.

Users often have multiple devices with different passkeys, so Relying Parties must support the association of multiple passkeys with one account. Passkey's user name and display name help the user select the appropriate one. They are set by the Relying Party and stored in the device. They can contain any information provided by the Relying Party, with the first usually being the name of the user's account, while the display name is the name of the person. These are not used for account matching; instead, the passkey's user handle, which is not presented to the user, is used to identify the user account.

To protect user privacy and prevent user tracking, user data in passkeys is limited to necessary information. Even when the user name and display name are personally identifying, data is communicated encrypted, saved locally, and displayed securely and only when necessary. Account-related information remains with the service. A Relying Party cannot discover credentials without user prompting, user devices are not identifiable by default, and Relying Parties must prevent information on existing accounts and passkeys from leaking through revealing responses.

4.1 Reauthentication

Reauthentication is a common situation where the user is already logged in but needs additional authentication because a session has expired or because the user wants to perform a sensitive operation, such as adding a shipping address or finalising a transaction. With password-based

authentication, the user is prompted to enter the password to reauthenticate. With passkeys, the application can simply prompt the user to unlock the device again.

4.2 Decommissioning of passkeys

If a passkey is lost or compromised, the user can inform the service to delete it from the list of their registered credentials. Otherwise, the user may simply delete a passkey from one or all devices without notifying the service. The service may later deregister the deleted or otherwise unused passkey due to inactivity.

4.3 Backup, loss and recovery

Passkeys are typically shared between multiple devices for synchronisation and backup. The biometric or PIN screen and the device's secure storage protect the passkey's private key on a lost or stolen device. If the user loses all devices, a new passkey must be created and registered from scratch. Syncing of passkeys is end-to-end encrypted, but some ecosystems may keep passkeys in encrypted form. Android and Apple ecosystems have their own methods for passkey recovery from these backups. Passkeys on a FIDO security key can also act as recovery credentials. If a user wants to switch between ecosystems, they can use the old device to sign in on the new device and then create a passkey for the new platform.

4.4 Applications development

Several resources and tutorials are available to implement passkeys in web and mobile applications, including those provided by Yubico, Google and Apple. Apple's tutorial covers both Apple WebKit and web HTML/JS programming. The WebAuthn.me debugger is useful in experimenting with passkey registration and authentication parameters. There are also a few authentication frameworks that facilitate the development of web applications that utilise passkeys.

4.5 Use of passkeys in MFA and LoA applications

Passkeys already provide a form of multifactor authentication by embodying two factors: the user devices they are kept on and knowledge or personal characteristics presented by biometrics or PIN. They can be used in MFA scenarios along with other independently acquired and verified factors. WebAuthn and FIDO2 attestations allow for the creation and use of credentials with specific characteristics such as a type of user verification, cryptographic algorithm, key lengths, authenticator device storage and authenticator connection protection, attestation and certification. However, passkeys introduce a sharing platform as a third party required for convenient use across devices, and the platform's end-to-end encryption and online backup for recovery purposes, which adds some uncertainty. In compliance-required environments, device-bound passkeys on FIDO security keys and the use of flags that indicate whether a credential can be or has been backed up could be a solution that has to be validated in practice. Regulatory regimes have yet to recognise passkeys as sanctioned multifactor authentication credentials.

5 Implementation for R&E

Since services that use passkeys rely on the WebAuthn API, a small change is required for those who already implement it. This mainly affects the UI for signing in, where the application needs to be updated to initiate a sign-in using browser support for passkeys.

- Federated usage – To enable passkey-based authentication, identity providers (IdPs) should implement passkeys in their login forms and set up passkey registration and management pages. No additional infrastructure is required for federated usage.
- Service integration – For services that use HTTP/HTTPS and do not rely on an external identity provider or federated login as a Relying Party, passkeys can be implemented in the same way as any passkey-aware web application.
- Proxy services – Identity providers may also develop passkey-aware proxy services responsible for passkey sign-in, registration and management, in the same way they would in a federated environment, but without the constraints imposed by identity federation rules or requirements. Instead, they can set up their own SAML or OpenID interfaces. It is recommended to add user-facing passkey support to the existing federated or standalone IdPs in a way that does not disrupt downstream services.
- Use with legacy services – If passkey support is added to an already used IdP, relying services should not be impacted in any way.
- Fragmentation across ecosystems – Passkeys are not shared across different operating system and browser ecosystems, so users need to create a new passkey for each ecosystem and service they use. This diminishes the user experience and reduces the user's control over their passkeys.
- Vendor independence – To avoid platform lock-in, it is essential to have independent passkey solutions that are widely adopted in the market and seamlessly integrate with various operating systems and browsers. Several password managers are planning to offer passkey integration as a standard feature in 2023.

Despite the weaknesses of passwords, they must remain available because not all users' devices have passkeys. Passkeys are becoming more integrated into the hardware and could become the primary means of accessing digital services, but there are challenges with updating identity providers to support them. The problem of fragmentation between passkey platforms is yet to be addressed. These platforms should increase transparency about what they do and the quality of used passkey credentials, authenticators and backups, and there are perception and trust issues to consider.

6 REFEDS profiles and passkeys

The REFEDS Assurance Framework (RAF) [https://refeds.org/assurance, https://wiki.refeds.org/display/ASS/REFEDS+Assurance+Framework+ver+1.0] provides organisations with comprehensive guidelines to establish and maintain trust in their online identities and services. The RAF covers both technical and non-technical requirements, including identifier uniqueness, identity assurance and attribute assurance. Since passkeys are a credentials technology, they are mostly unrelated to the RAF. However, during passkey registration, organisations must ensure that the RAF requirements are met at the beginning of the registration process when the user's identity is established through means other than the passkey. The RAF also determines when a credential needs to be renewed, replaced, or revoked. For passkeys, this is achieved or triggered by the deletion of the Relying Party's record for the passkey in question.

REFEDS has developed profiles for single and multi-factor authentication, which do not currently cover passkeys explicitly but can be met by using the features defined in the WebAuthn specification. The detailed report outlines how to implement passkeys in a way that meets the single-factor authentication profile, and the requirements that must be fulfilled. These requirements include a minimum secret length, maximum lifespan and cryptographic protection in transit and at rest. Additionally, the replacement of a lost authentication factor must follow specific procedures that do not rely solely on knowledge-based authentication.